REMARKS

The Examiner has rejected claims 1-18 as anticipated by Bellare. Applicants respectfully traverse this rejection and beg for reconsideration in view of the remarks set out below.

The Bellare reference

Bellare has been cited as disclosing certain hash functions and MAC functions.

At page 8, Bellare has been cited as disclosing the keyed hash function $F_k(x_1, x_2, \ldots, x_n)$, which is diagrammed on the page attached hereto as an Appendix. In the diagram, *f* designates a (single iteration) compression function. At page 9, Bellare has been cited as disclosing the MAC function $NMAC_k(x)$, where $x = (x_1, x_2, \ldots, x_n)$. An example of the NMAC function is also diagrammed on the attached Appendix. At page 13, Bellare is cited as disclosing the MAC function $HMAC_k(x)$. An example of HMAC is also diagrammed on the attached Appendix. The functions NMAC and HMAC are also diagrammed in Figures 7 and 8, respectively, of the instant Specification. (The representation of HMAC in the attached Appendix differs somwhat, in a detail of how the keys are input, from the representation in Figure 8 of the Specification. However, that difference does not affect the arguments that follow.)

In regard to $F_k(x_1, x_2, \ldots, x_n)$, it will be seen that if the input message *x* is longer than one block length, the hash function will be carried out (at least in the example shown) by iterating the compression function *f*. Significantly, the key *k* is applied only to the initial iteration. Thus, $F_k(x_1, x_2, \ldots, x_n)$ is—in and of itself—a keyed hash function without any further hash function "nested within [the] keyed hash function" $F_k(x_1, x_2, \ldots, x_n)$.

In regard to $NMAC_k(x)$, it will be seen that even if the input message *x* fits within a single block length, the compression function *f* (at least in the example shown) must be called at least twice. The first iteration, as shown in the diagram,

is the iteration that takes k_2 as an input key, and the second or last iteration is that which takes k_1 as an input key.

In regard to $HMAC_k(x)$, it will be seen that even if the input message *x* fits within a single block length, the compression function *f* (at least in the example shown) must be called at least four times: twice taking initial variable *IV* and a padded key *k* as input, at least once taking at least a portion of the message as input, and once taking the hashed message and a key derived from *k* and *IV* as input.

Claim 1 and claim 14

Claim 1, as currently amended, is drawn to a method of processing a message for authentication. The claimed method includes a step of determining whether the message *x* fits within an input block of the compression function. If *x* does fit within an input block, a result of the compression function goes to produce a message authentication code (i.e., a MAC) without further iteration of the compression function. As a consequence, computational efficiency is increased while maintaining a desirable level of security.

Bellare fails to teach or suggest any MAC-generation method in which the compression function is called only once. The MAC-generation algorithms which Bellare describes are NMAC and HMAC. As explained above, NMAC requires at least two iterations, and HMAC requires at least four, even when the input message fits within a single input block.

Claim 1 contains the further limitation that the input message is processed by a hash function nested within a keyed hash function. As explained above, the function $F_k(x_1, x_2, \ldots, x_n)$ as described in Bellare does not anticipate or suggest this limitation, because it is simply a keyed hash function and has no further hash function nested within it. To the extent that any earlier iteration might be considered "nested" within a later iteration, it should be noted that it is the earlier iteration that has been "keyed", whereas there is no possibility of adding a further key to the later iterations. Thus, it would not be accurate to

construe $F_k(x_1, x_2, ..., x_n)$ as containing any function nested within a keyed hash function.

Thus, each of the functions NMAC, HMAC, and $F_k(x_1, x_2, ..., x_n)$ as described in Bellare is inconsistent with at least some feature of the method recited in claim 1, as currently amended.

Moreover, amended claim 1 is drawn to a method which leads to production of a message authentication code (MAC), and which includes a step of determining whether the input message fits within a block so as to choose between alternative subseqent process steps, i.e., between processing with a single call to a compression function, versus processing with a hash function nested within a keyed hash function, which necessitates at least two calls to a compression function. Bellare fails to teach, or even to suggest, any function for MAC generation that has two branches, leading to different types of processing, depending on a determination of whether the input message fits within a block.

For the above reasons, Applicants respectfully submit that claim 1 is neither anticipated by Bellare, nor obvious over Bellare. Because currently pending claims 2-4 depend from claim 1, and because amended apparatus claim 14 incorporates, *mutatis mutandis*, substantially the same limitations as amended claim 1, it is respectfully submitted that these claims are likewise novel and non-obvious over Bellare.

Claim 7 and claim 16

Claim 7 as currently amended is drawn to a method of processing an input message, in which a "first portion" of the input message is hashed. A second portion of the input message is combined with the hashed first portion, and the combination is used as input to a keyed hash function. The combination is formed by concatenating the second portion with the hashed first portion.

Bellare fails to teach, or even to suggest, any hash function or MAC-generating function in which two portions of an input message are processed differently, and then concatenated, and the concatenation used as input for further processing. Therefore, it is respectfully submitted that claim 7 is neither

anticipated by Bellare, nor obvious over Bellare. Because currently pending claims 8-11 depend from claim 7, and because currently amended apparatus claim 16 incorporates, *mutatis mutandis*, substantially the same limitations as claim 7, Applicants respectfully submit that these claims are likewise novel and non-obvious over Bellare.

Claim 19 and claim 21

New claim 19 is drawn to a method of processing a message x for use in a message authentication scheme. The method includes a conditional step, which comprises using a hash function to compress at least a portion of x, on condition that x exceeds a block size. The output of the conditional step is an intermediate result. The input message x (if it is smaller than a block size), or the intermediate result (if the conditional step has been invoked) is then compressed using a keyed compression function.

For substantially the same reasons set out above in regard to claim 1, Applicants submit that the function $F_k(x_1, x_2, ..., x_n)$ as described in Bellare is inconsistent with claim 19, at least when x is larger than one block size. That is, if x is larger than one block size, the early iterations of the compression function f might *arguendo* be described as compressing a "portion" of x, but the compressed portion is not an "intermediate result" or even part of an "intermediate result" in the sense of claim 19. In particular, that portion of x that has been compressed by the early iterations of the compression function is not thereafter compressed *with a keyed compression function* as recited in the claim.

Applicants also submit that the functions NMAC and HMAC as described in Bellare are inconsistent with the method of claim 19. That is, claim 19 requires that x (in the event that the input message is smaller than a block size) or the intermediate result y (if the input message is longer than a block size) be compressed with a keyed compression function. Thus, if x is shorter than a block size, it passes directly to the keyed compression function, whereas if x is longer than a block size, it is compressed and then passed to the same keyed compression function. NMAC and HMAC might *arguendo* be described as

including, as the last processing step, a keyed compression function. However, both NMAC and HMAC require that no matter how short the input message is, it must be processed by at least one call to the compression function (shown in the attached diagram as the first compression operation in NMAC, and the second compression operation in the "top line" of HMAC) before being passed to the final compression operation. Therefore, both NMAC and HMAC are inconsistent with Step (b) of claim 19, i.e., with "compressing x or y with a keyed compression function".

Moreover, as explained above, Bellare neither teaches nor suggests any MAC-generating function that contains any conditional operations.

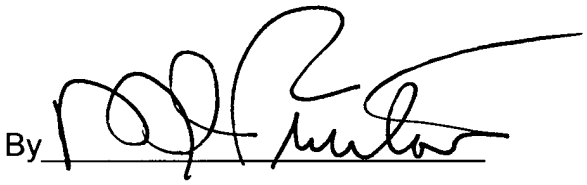
Accordingly, it is respectfully submitted that claim 19 is neither anticipated nor obvious over Bellare. Because new claim 20 depends from claim 19, and because new apparatus claim 21 contains, *mutatis mutandis* substantially the same limitations as claim 19, it is further submitted that these claims are likewise novel and non-obvious over Bellare.

Conclusion

Having responded to all points of rejection, Applicants beg for reconsideration in view of the preceding remarks, leading to allowance of all claims now pending.

Respectfully submitted,

Sarvar Patel

By 

Martin I. Finston, Attorney
Reg. No. 31613
908-582-2908.

Attachments
Appendix

Date: July 5, 2005

Docket Administrator (Room 3J-219)
Lucent Technologies Inc.
101 Crawfords Corner Road
Holmdel, NJ 07733-3030

APPENDIX